TEMA 1. Introducción

Física Computacional I

Carolina Mendoza

Escuela Politécnica Superior
Universidad Nebrija

TEMA 1. Introducción

Lenguajes de programación

- 1840s. Ada Lovelace, George Babbage (algunos programas complejos)
- 1880s. Herman Hollerith (datos sobre medios físicos)
- 1940s. Von Neumann (arquitectura que permite guardar programas en una memoria)
- 1949. Lenguaje ensamblador (remplaza a las instrucciones máquina específicas, formato de texto)
- 1947. Grace Hopper (debugging de códigos)
- 1949. Grace Hopper (primer compilador)

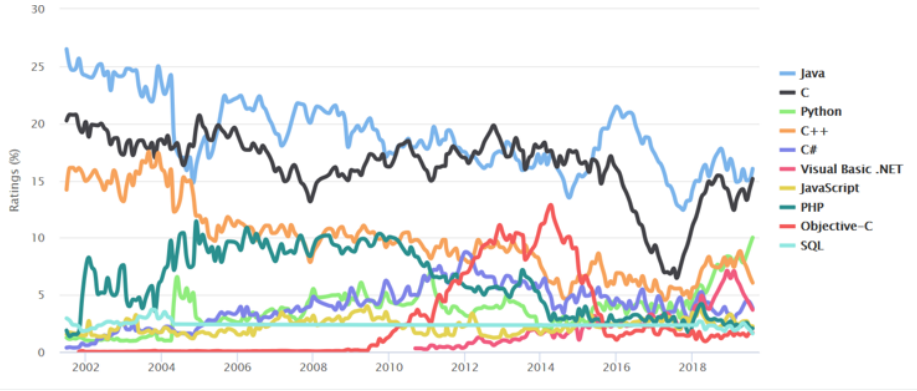
TEMA 1. Introducción

Lenguajes de programación

- 1954. Fortran (primer lenguaje de alto nivel) (John Backus en IBM) . Primer lenguaje moderno.
- 1971. Lenguaje C
- 1983. Lenguaje C++
- 1989. Python
- 1991. HTML
- 1995. Java

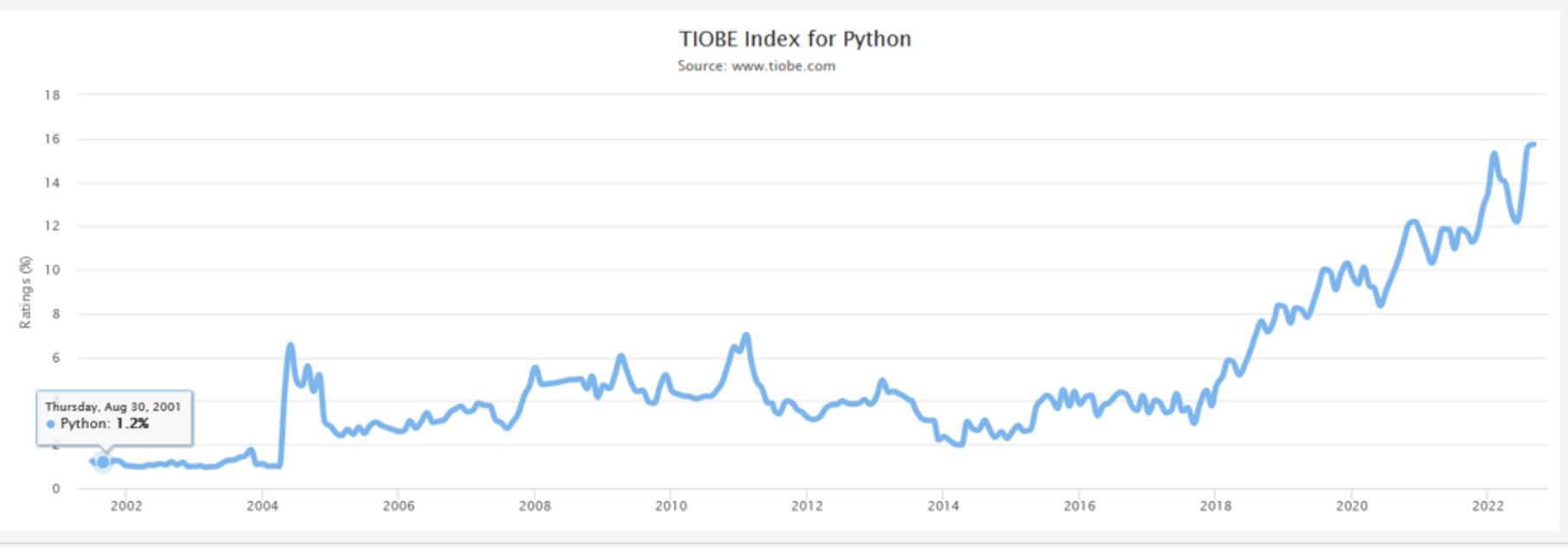
TEMA 1. Introducción

Lenguajes de programación



TEMA 1. Introducción

Lenguajes de programación



TEMA 1. Introducción

Lenguajes de programación

python™

- Python: Lenguaje de programación dinámico que soporta varios paradigmas de paradigmas de programación:
  - Programación procedimental
  - Programación orientada a objetos
- Estándar: El código de bytes de Python se ejecuta en el intérprete de Python (similar a Java)
- Código independiente de la plataforma

TEMA 1. Introducción

Python: historia

- Inicio: diciembre de 1989 por Guido van Rossum
- 16.10.2000: Python 2.0
- Soporte de Unicode
- Colector de basura
- Proceso de desarrollo más orientado a la comunidad
- 3.12.2008: Python 3.0
  - No es 100% compatible con versiones anteriores
- 2007 y 2010 lenguaje de programación más popular (Índice TIOBE)
- Recomendado para la programación científica (Nature News, NPG, 2015)
- Versión actual: Python 3.9.2

UNIVERSIDAD NEBRIJA

Lo bueno

- Fácil de leer/escribir
- Muy usado (gratis)
- Fácil iniciarse
- Similar a Matlab/Octave (no tan amigable ni robusto)
- Base de entornos de Deep Learning: PyTorch, Tensorflow
- Front-end de numeroso software
- Gran comunidad

UNIVERSIDAD NEBRIJA

Lo malo

- Múltiples versiones de Python
- Mala (pésima!) gestión de paquetes y dependencias
- A menudo diferentes proyectos requieren versiones diferentes de los mismos paquetes
- Muy lento (Algoritmos de cálculo intensivo: Fortran, C, C++ son más rápidos)
- Indexación no intuitiva

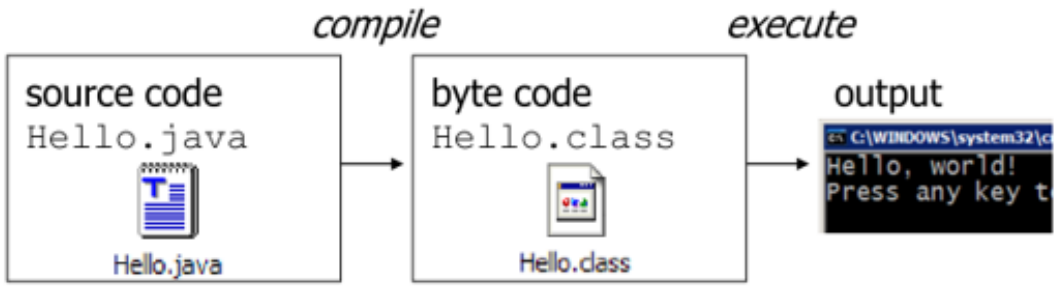
UNIVERSIDAD NEBRIJA

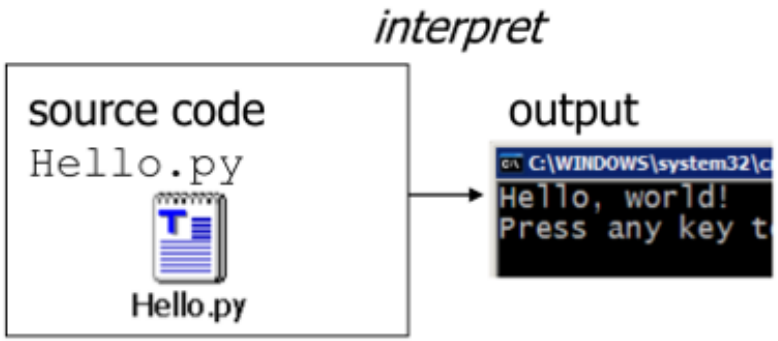
El parche

- Mantener los proyectos en entornos distintos y aislados
- Cada entorno puede usar distintas versiones de Python
- Cada entorno contiene sus propios paquetes (con sus versiones)

TEMA 1. Introducción

Lenguaje interpretado





TEMA 1. Introducción

Características del lenguaje

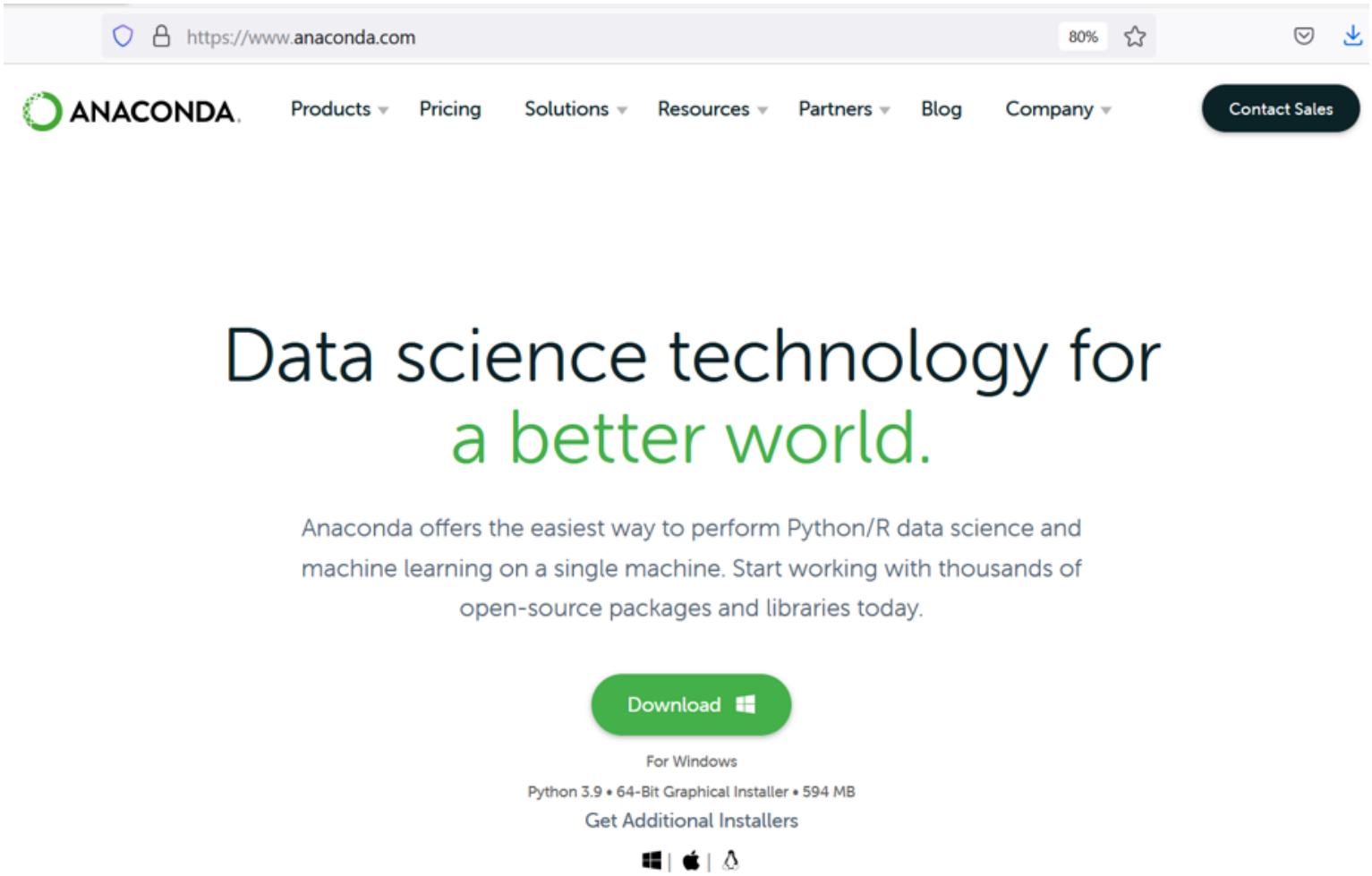
- Fuertemente tipado: el intérprete siempre respeta los tipos de cada variable (una cadena es siempre una cadena, un entero siempre es un entero ,etc)
- Dinámicamente tipado: no hay declaración de variables, las variables pueden ser reasignadas a otro tipo
- Python es un lenguaje interpretado
- Python es lento, pero a cambio puede correr librerías numéricas muy optimizadas de C/C++ que permiten cálculo científico rápido
- Los bloques de código se crean mediante sangrados, en lugar de corchetes como en C++
- Los sangrados son de dos o cuatro espacios. Ser consistente en el código

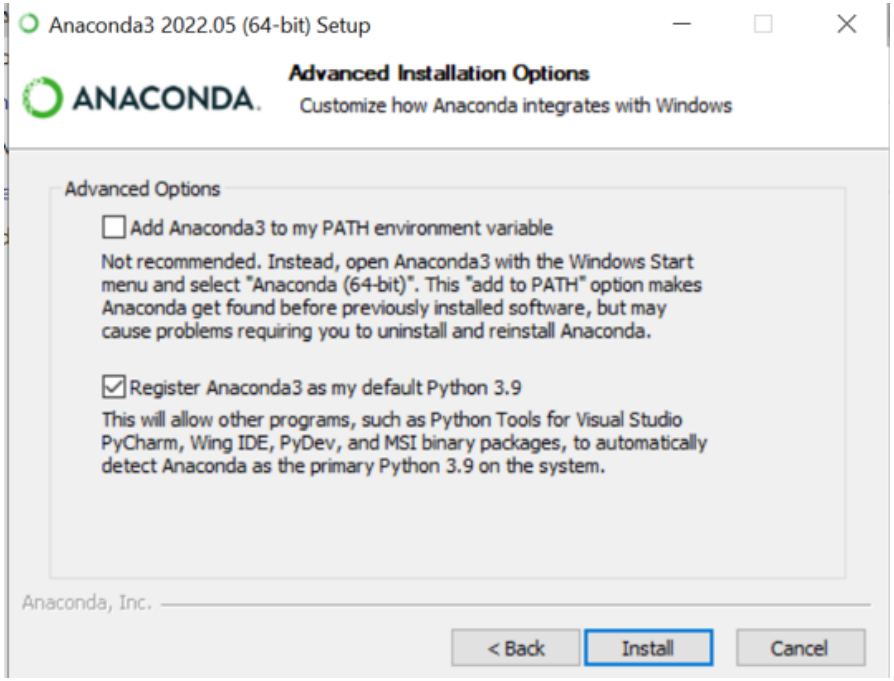
Anaconda

- Entorno muy bien mantenido y robusto
- Cross-platform: Windows, Linux, MacOS
- Permite crear y manejar entornos con facilidad
- https://www.anaconda.com/

TEMA 1. Introducción

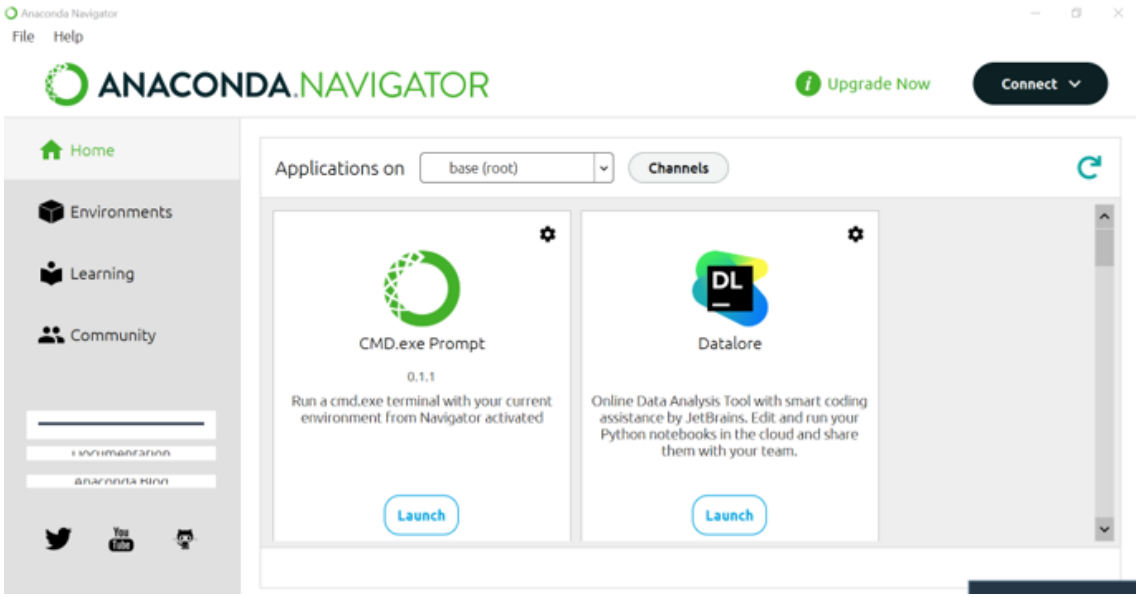
Entorno




Anaconda3 2022.05 (64-bit) Setup
ANACONDA.
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Add Anaconda3 to my PATH environment variable
Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.
Register Anaconda3 as my default Python 3.9
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.9 on the system.
Anaconda, Inc.
< Back
Install
Cancel

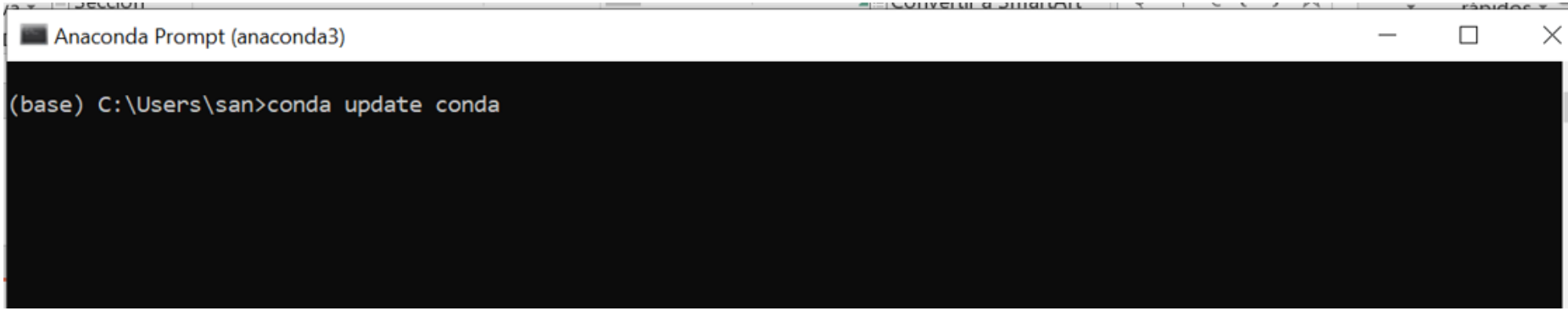
TEMA 1. Introducción

Entorno



TEMA 1. Introducción

Entorno



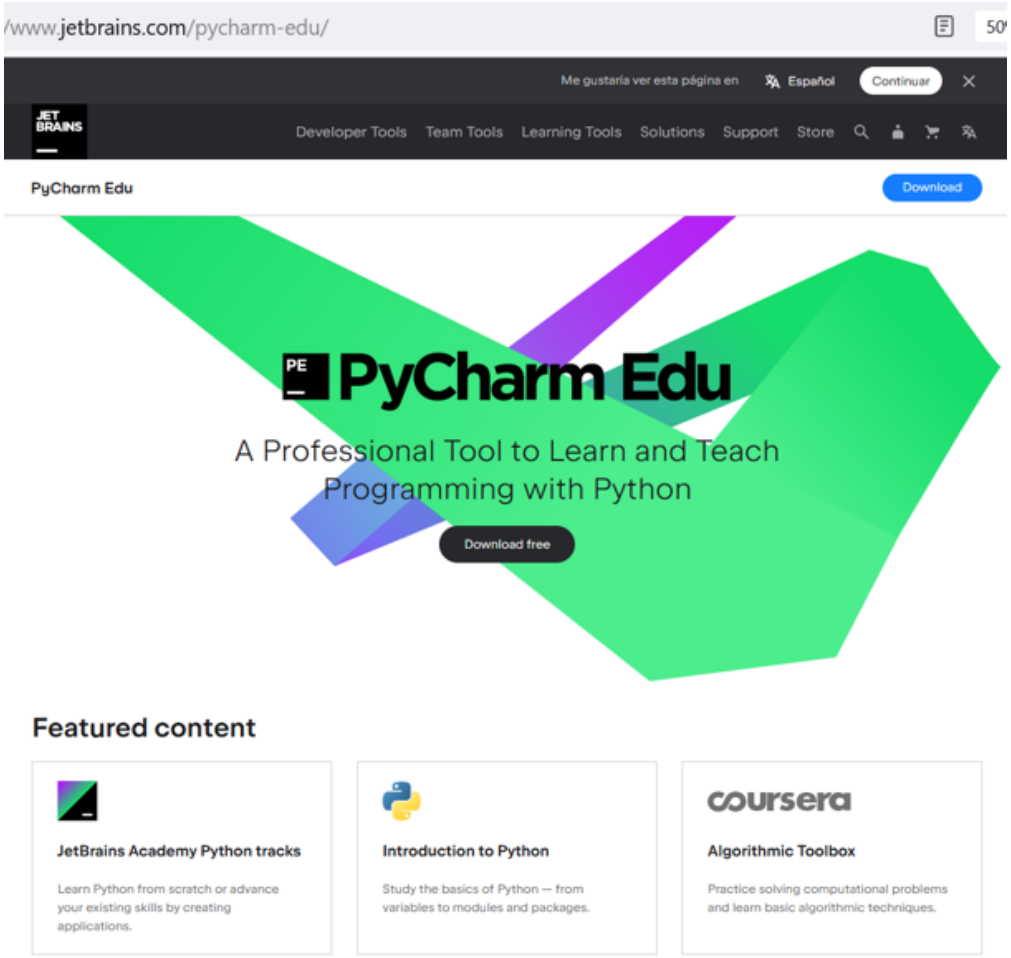
TEMA 1. Introducción

Editores

- PyCharm https://www.jetbrains.com/pycharm-edu/
- Spyder
- Visual Studio Code
- Sublime Test
- Atom
- Vim
- Emacs
- Notepad++

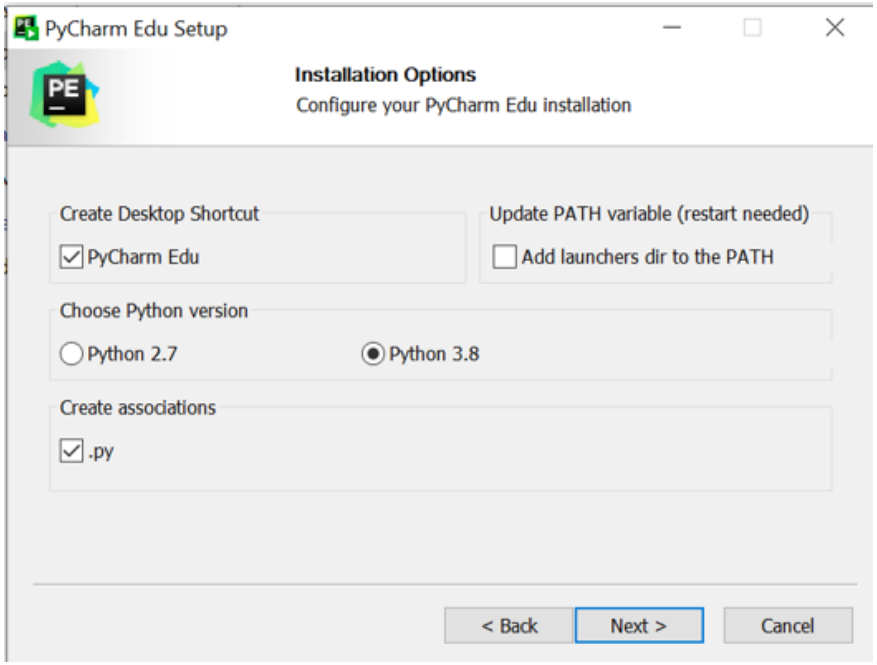
TEMA 1. Introducción

PyCharm



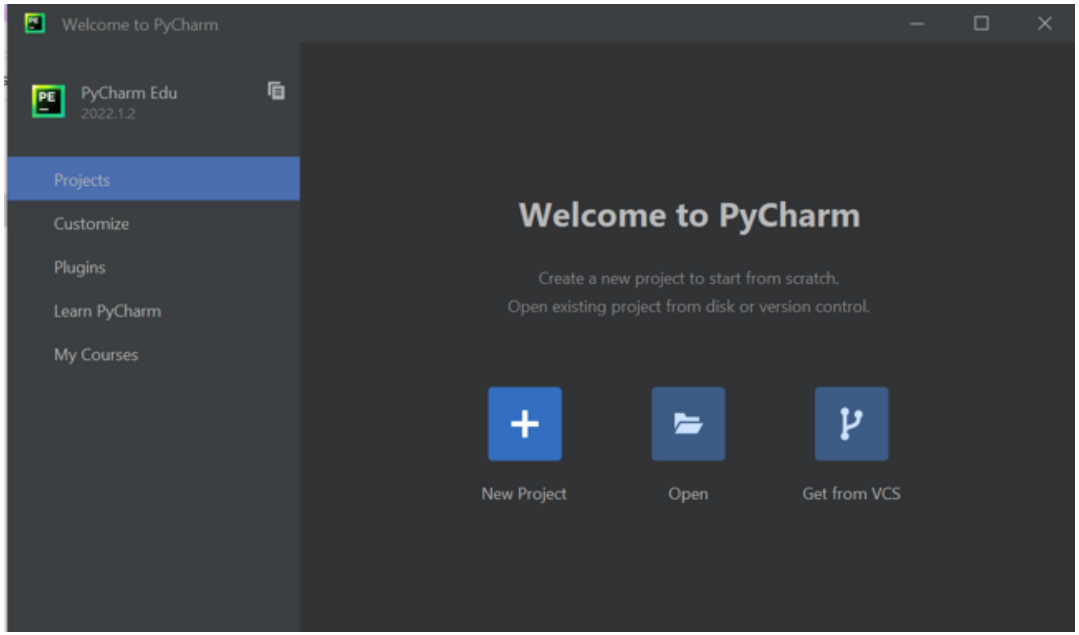
TEMA 1. Introducción

PyCharm



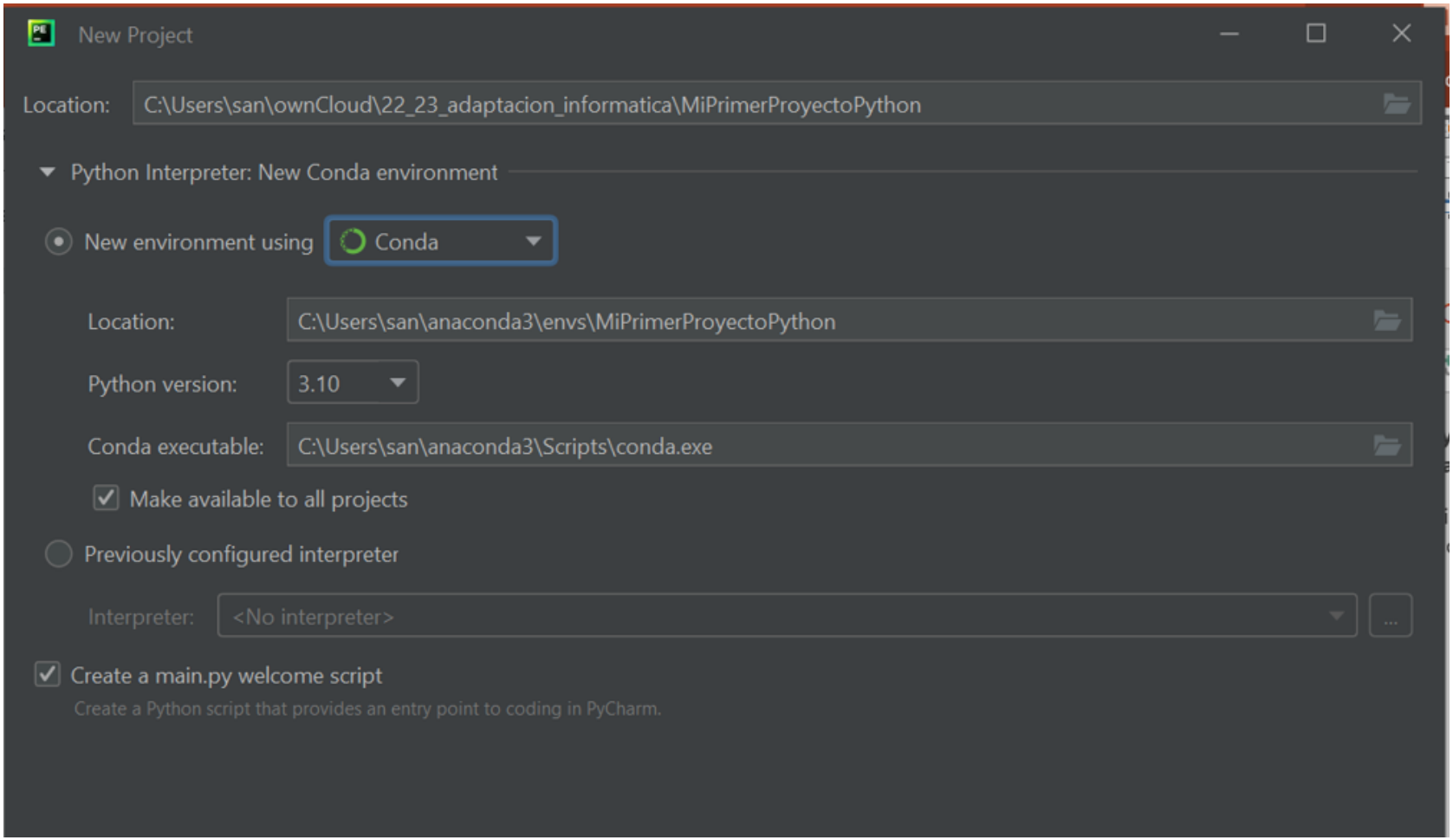
TEMA 1. Introducción

PyCharm



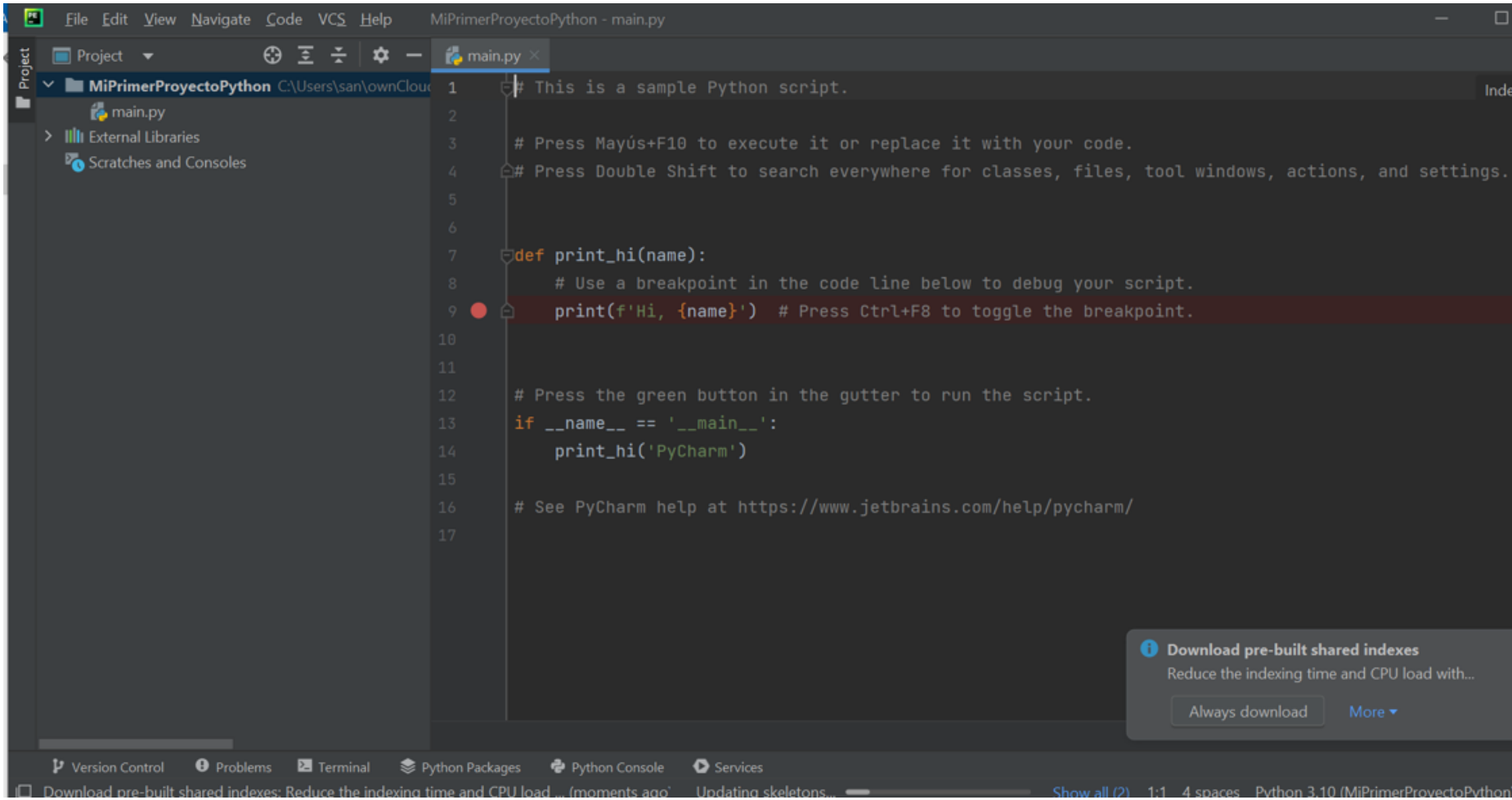
TEMA 1. Introducción

PyCharm



TEMA 1. Introducción

PyCharm


File Edit View Navigate Code VCS Help
MiPrimerProyectoPython - main.py
Project
main.py
MiPrimerProyectoPython C:\Users\san\ownCloud
main.py
External Libraries
Scratches and Consoles
# This is a sample Python script.
# Press Mayús+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
def print_hi(name):
# Use a breakpoint in the code line below to debug your script.
print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.
# Press the green button in the gutter to run the script.
if __name__ == '__main__':
print_hi('PyCharm')
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
Download pre-built shared indexes
Reduce the indexing time and CPU load with...
Always download
More
Version Control
Problems
Terminal
Python Packages
Python Console
Services
Python 3.10 (MiPrimerProyectoPython